SOFIA's Choice: An AI Approach to Scheduling Airborne Astronomy Observations

Jeremy Frank and **Michael A. K. Gross**[*] and **Elif Kürklü**[†]
NASA Ames Research Center
Mail Stop N269-3
Moffett Field CA 94035-1000
{frank,ekurklu}@email.arc.nasa.gov, mgross@mail.arc.nasa.gov

Abstract

We describe an innovative solution to the problem of scheduling astronomy observations for the Stratospheric Observatory for Infrared Astronomy, an airborne observatory. The problem contains complex constraints relating the feasibility of an astronomical observation to the position and time at which the observation begins, telescope elevation limits and available fuel. Solving the problem requires making discrete choices (e.g. selection and sequencing of observations) and continuous ones (e.g. takeoff time and setting up observations by repositioning the aircraft). The problem also includes optimization criteria such as maximizing observing time while simultaneously minimizing total flight time. We describe a method to search for good flight plans that satisfy all constraints. This novel approach combines heuristic search, biased stochastic sampling, continuous optimization techniques, and well-founded approximations that eliminate feasible solutions but greatly reduce computation time.

Introduction

The Stratospheric Observatory for Infrared Astronomy (SOFIA) is NASA's next generation airborne astronomical observatory. The facility consists of a 747-SP modified to accommodate a 2.5 meter telescope. SOFIA is expected to fly an average of 140 science flights per year over it's 20 year lifetime, and will commence operations in early 2005. The SOFIA telescope is mounted aft of the wings on the port side of the aircraft and is articulated through a range of 20° to 60° of elevation. The telescope has minimal lateral flexibility; thus, the aircraft must turn constantly to maintain the telescope's focus on an object during observations. A significant problem in future SOFIA operations is that of scheduling Facility Instrument (FI) flights in support of the SOFIA General Investigator (GI) program. GIs are expected to propose small numbers of observations, and many observations must be grouped together to make up single flights. Approximately 70 GI flight per year are expected, with 5-15 observations per flight.

The scope of the flight planning problem for supporting GI observations with the anticipated flight rate for SOFIA makes the manual approach for flight planning daunting. There has been considerable success in automating observation scheduling for ground-based telescopes (Bresina 1996), space-based telescopes such as Hubble Space Telescope (Johnston & Miller 1994), Earth Observing Satellites (Potter & Gasch 1998) and planetary rovers (Smith 2004). However, the SOFIA flight planning problem differs from these problems in a variety of ways. Observations are feasible over large, continuous regions of space and time. The motion of the aircraft is governed by differential equations, and the aircraft can be flown in any direction for any length of time to enable an observation. The principal feasibility condition for observations is a nonlinear function over the solution to the equations of motion. As a consequence of these factors, even though SOFIA has a "closed tour" constraint that makes it appear similar to problems such as the Traveling Salesperson Problem, there are no fixed waypoints to define routes. Also, the SOFIA problem cannot be characterized only by discrete decisions. The complexity of the differential equations and feasibility constraints makes it difficult to find good heuristics, and the expense of solving the differential equations impacts solver performance.

In this paper, we describe a combination of heuristic search, biased stochastic sampling, approximations and continuous optimization methods to produce an algorithm that efficiently finds good solutions to the SOFIA flight planning problem. The rest of the paper is organized as follows. We first formally describe the problem of planning GI flights, the constraints on flight plans, and the optimization criteria used to compare valid flight plans. We then briefly describe the search algorithm used to construct good flight plans, which combines heuristic search with stochastic sampling. Initial experiments with this algorithm indicate that it spends much of its time deciding what observation to schedule next. We then describe a combination of well-founded approximations and continuous optimization techniques that allow us to eliminate a large number of expensive computations while still finding good flights. We describe experiments to validate the approach. Finally, we conclude and discuss future work.

SOFIA's Choice

The SFPP (Single Flight Planning Problem) consists of a number of observation requests, a flight day, and a takeoff and landing airport. The objective is to find a flight plan that maximizes the summed priority of the observations performed while obeying the constraints governing legal flights. The aircraft can perform two different classes of activities during a flight. *Flight-legs* require tracking an object and

[*]Universities Space Research Association
[†]QSS Group, Inc.

are only legal if the object is within the telescope elevation limits throughout the observation. *Dead-legs* can be used to reposition the aircraft to enable flight-legs, but no observations are performed. A distinguished class of dead-legs are used to take off and return to the landing airport.

The input to the SFPP consists of a set of observation requests, each consisting of the Right Ascension (RA) and Declination (Dec), observation duration, priority, earliest start time and latest end time; a flight date; initial fuel load; earliest takeoff time and latest landing times; and the designated takeoff and landing airports (which need not be the same). The primary objective is to find a flight plan that maximizes the summed priority of the observations of the observations performed. A secondary criteria is to maximize efficiency (the proportion of the flight spent performing observations). Since it is intractable to find the best possible plan, we limit ourselves to searching for *good* plans that perform many observations of high priority. Solving the SFPP requires selecting the set of observations to service, ordering them and inserting necessary dead-legs.

Unlike traditional scheduling problems studied in the AI and OR communities, the principle constraint links the position of the aircraft and the time an observation begins to the telescope elevation required to see the object. There are few absolute temporal constraints on individual observations and also few relative temporal constraints between observations. However, there are numerous combinations of positions and times at which observations are infeasible. While it is possible to make observations feasible by repositioning the aircraft or delaying the observation, poorly chosen orderings can lead to inefficient flights with few scheduled observations.

Constraints on Valid Flights

In this section we elaborate on the various constraints on valid solutions to the SFPP. The constraints linking aircraft motion and observation feasibility are the most important component of the problem, so we describe them in detail here. (This description differs from that in previous work (Frank & Kürklü 2003) in that it is simpler and more accurate.) If an observation is scheduled, then it must be performed for the requested duration without interruption. As we will see, the elevation depends on the coordinates of the object being observed, the position of the aircraft, and the time. SOFIA can view objects between 20° and 60° of elevation; checking this constraint requires first computing the aircraft's ground track throughout the course of the observation. Figure 1 shows the interaction between the object's position in the sky at a particular time, the aircraft's ground track, and the telescope elevation. The Earth is modeled as an oblate spheroid $\mathbf{E}$, whose surface is defined by the equation $\frac{x^2}{a^2} + \frac{y^2}{a^2} + \frac{z^2}{c^2} = 1$ where $c < a$.

Let $\mathbf{p}$ be the aircraft's current position, and θ be the (Sidereal) time that the aircraft is at $\mathbf{p}$. Let $\vec{\mathbf{S}}$ be the vector from the center of $\mathbf{E}$ to $\mathbf{p}$. Let $\vec{\mathbf{T}}$ be the vector defining the vector to an astronomical object o, and $\mathbf{P}$ as the plane tangent to $\mathbf{E}$ at $\mathbf{p}$. Let $\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$ be the unit vectors in the x, y, z directions respectively. Let $\vec{\mathbf{N}}$ be the vector normal to $\mathbf{P}$: $\vec{\mathbf{N}} = \frac{p_x}{a^2}\hat{\mathbf{i}} + \frac{p_y}{a^2}\hat{\mathbf{j}} + \frac{p_z}{c^2}\hat{\mathbf{k}}$ (Note that $\vec{\mathbf{S}}$ and $\vec{\mathbf{N}}$ are generally not parallel since $\mathbf{E}$ is a spheroid.) Let $\vec{\mathbf{T_P}}$ be the projection of $\vec{\mathbf{T}}$ onto $\mathbf{P}$; this is the *object azimuth* at $\mathbf{p}$, and is given by

$$\vec{\mathbf{T_P}} = \vec{\mathbf{T}} - \frac{\vec{\mathbf{T}}\vec{\mathbf{N}}}{||\vec{\mathbf{N}}||^2}\vec{\mathbf{N}} \tag{1}$$

Let $\vec{\mathbf{V}}$ be the desired heading of the aircraft. The observatory must track the object inducing $\vec{\mathbf{T}}$, subject to the constraint that the angle between $\vec{\mathbf{V}}$ and $\vec{\mathbf{T_P}}$ is 270°, because the telescope points out the left-hand side of the aircraft. Let $\mathbf{R}_{\vec{\mathbf{N}}}(\mathbf{270^\circ})$ be a rotation matrix that rotates a vector 270° around $\vec{\mathbf{N}}$, and v be the airspeed of the aircraft; then

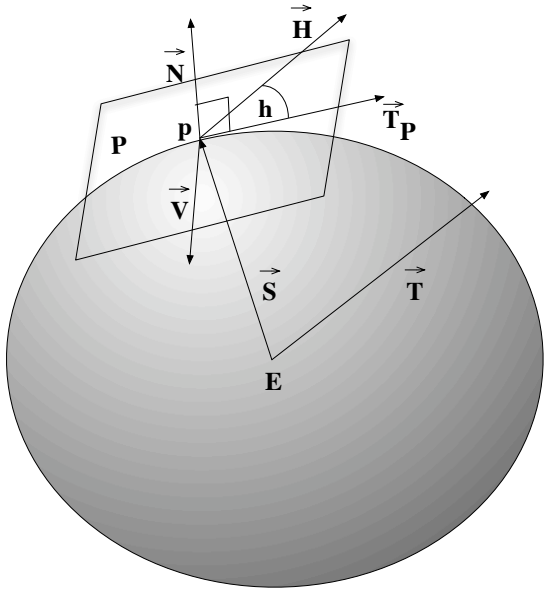

Figure 1: The Cartesian formulation of the instantaneous equations of motion of the aircraft and the elevation. We have exaggerated the spheroid E.

$$\vec{\mathbf{V}} = v\mathbf{R}_{\vec{\mathbf{N}}}(\mathbf{270^\circ})\frac{\vec{\mathbf{T_P}}}{||\vec{\mathbf{T_P}}||} \tag{2}$$

Let $\vec{\mathbf{H}}$ be the elevation vector with respect to $\mathbf{P}$. We also require the angle h between $\vec{\mathbf{H}}$ and $\vec{\mathbf{T_P}}$ obey the constraint $20^\circ \leq h \leq 60^\circ$ throughout an observation. Most targets are sufficiently far from Earth that we can assume $\vec{\mathbf{H}} = \vec{\mathbf{T}} + \vec{\mathbf{S}}$ From vector calculus we then get the equation for the elevation h:

$$h = \cos^{-1}\left(\frac{\vec{\mathbf{H}}\vec{\mathbf{T_P}}}{||\vec{\mathbf{H}}||\,||\vec{\mathbf{T_P}}||}\right) \tag{3}$$

$\vec{\mathbf{T}}$ is a function of o and θ; this is because the Earth rotates on its axis. The vector $\vec{\mathbf{T}}$ traces a circle of radius $x^2 + y^2 =$

$\frac{c^2-d}{c^2}$, where $d = |\frac{\delta}{90^\circ}|$ in 24 hours (see (Meeus 1991) for an explanation of this).

The instantaneous change in $\mathbf{p}$ as the aircraft tracks o is $\frac{d\mathbf{p}}{d\theta} = \vec{\mathbf{V}}$. Since $\vec{\mathbf{V}}$ is a function of $\vec{\mathbf{T}}$, it is a function of $o, \mathbf{p}$ and θ. Solving for the ground track is necessary to compute h and check the elevation constraints. It is worth noting that this formulation also makes it easy to add the effect of winds by adding the appropriate vectors to $\vec{\mathbf{V}}$, and also correct for aircraft pitch by rotating about $\vec{\mathbf{V}} \times \vec{\mathbf{N}}$, but we omit these for brevity. The ground track and elevation constraints are solved using 5^{th}-order Runge-Kutta (Ferziger 1981) with error-adaptive step sizing.

The telescope is carried aboard a Boeing 747-SP aircraft. The fuel consumption of each engine depends on the aircraft weight, outside air temperature, drag, initial altitude and final altitude. The fuel consumption constraints are represented in a lookup table provided by Boeing; space precludes describing the fuel consumption constraint in more detail. A gridded wind and temperature model is available to correct the ground track in the face of winds and provide data for calculating fuel consumption.

ForwardPlanner()
F is (initially empty) current flight plan
Select takeoff time
while not done
  # E is (initially empty) set of feasible observations
  for each unscheduled observation o
    if Feasible(o, F)
      v=Evaluate(o, F)
      Add (o, v) to E
  end **for**
  if E is not empty
    Use values v to select e from E
    Extend F by e; empty E
  else done
return F
end

Figure 2: A sketch of the SFPP Flight Planning Algorithm.

Algorithm Description

Figure 2 describes ForwardPlanner, an algorithm for solving the SFPP (previously described in (Frank & Kürklü 2003)). ForwardPlanner combines progression based search, heuristics and stochastic sampling, resulting in a fast, incomplete randomized algorithm. An observation o is considered feasible at time θ and position $\mathbf{p}$ if there is a *dead-leg* of duration $\leq D$ after which the observation stays within the elevation limits for the required duration of the observation, and the aircraft can fly to the landing airport after the observation is finished. (D is an operational constraint, and is not strictly speaking an algorithm parameter.) The function Feasible() performs a search for the *shortest dead-leg* that satisfies these conditions. Each feasible observation o is evaluated by constructing a short extension to the flight plan; this is performed by Evaluate(), as shown in Figure 3. After adding o to the flight plan, up to K additional observations are added to the flight plan. This "lookahead" is performed

Evaluate(o, F**)**
K is the maximum extension
Extend F by o
repeat K times
  # L is the (initially empty) set of feasible observations
  for each unscheduled observation q not in F
    if Feasible(q)
      w=value of extending F by q
      Add (q, w) to L
  end for
  Use values w to select f from L
  Extend F by f; empty L
end repeat
v is value of F
return v

Figure 3: A sketch of the Evaluation method of the Forward-Planner Algorithm. The flight plan is extended by an observation o, the best possible flight plan possible after adding o is approximated by adding up to K additional observations.

to estimate the best flight plan possible after adding observation o. These short extensions are evaluated to estimate the value of the best plan conditioned on adding the observation o. When ForwardPlanner() decides how to extend a flight plan or Evaluate() conducts its lookahead, the candidates are evaluated using a heuristic. This heuristic is a weighted sum of the *priority* of the observations performed so far, the *efficiency* (ratio of time spent observing to total flight time) of the (incomplete) flight, the estimated time to return to the designated landing airport, and the total time spent in turns. (Details on the heuristics can be found in (Frank & Kürklü 2003)). The heuristic rank of each observation is treated as the mass of a probability distribution used to select the next observation. This technique is similar to Heuristic Biased Stochastic Sampling (HBSS), a technique used for scheduling ground based telescopes (Bresina 1996). This means that the "best" candidate need not be selected at any stage of the process, but has the highest probability of being selected next. This has proved to be an effective strategy when using inexpensive but somewhat inaccurate heuristics. The process of evaluating the feasible observations and adding the next observation to a flight is shown in Figure 4.

The principal cost of this algorithm is in the calls to Feasible(), where many flight-legs and dead-legs are constructed to test the conditions on object elevation. Let N be the number of observation requests, let K be the lookahead depth, and let M be the maximum number of observations that can be in any flight plan. Then the algorithm makes $O(N^2KM)$ calls to Feasible(); a proof of this appears in (Frank & Kürklü 2003).

Improving Algorithm Performance

Stochastic sampling approaches like the one employed in ForwardPlanner typically require many trials to find a good solution. The faster each trial is, the faster a good solution can be found. The predictive power of the heuristics is also

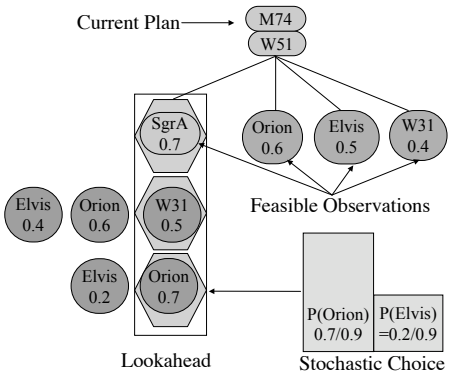

Figure 4: ForwardPlanner's search. At each step, all feasible observations are considered as the next observation in the plan. For each feasible observation, an extension of the plan is built using lookahead. The extensions are evaluated to determine which observation to perform next (the numbers inside each feasible observation.) The values are used to construct a probability distribution used to choose the next observation; each choice is indicated by a hexagon.

important, but not the focus of this paper. Our investigation into the first version of ForwardPlanner algorithm revealed that we spend a considerable amount of time deciding which observations are feasible, and most of this feasibility testing is performed in the lookahead phase of Evaluate() (in Figure 3). Feasibility testing is done by performing a brute force search for a short dead-leg. In the worst case, this requires ForwardPlanner to construct a very large number of legs; a typical number is $500,000$, evenly split between flight-legs and dead-legs. This is true even though the dead-leg duration is discretized and limited, as are the heading choices for the enabling dead-leg.

In this section we describe how to change the solution methodology to reduce the cost of checking feasibility without sacrificing performance. First we describe a modification to the ForwardPlanner that *restricts* the set of plans that can be built, increasing speed with only a small impact on the value of the flight plans found. We then show how to replace the expensive brute force search for dead-legs by a continuous optimization problem whose solutions are the desired dead-legs, again increasing speed with minimal performance impact.

Restricting the Set of Plans

The call to Feasible() may require a large number of expensive checks to ensure that the aircraft can return to the landing airport. In some cases, a short dead-leg enabling an observation makes it impossible to return home, while a longer dead-leg both enables the observation and allows the aircraft to return to the landing airport. However, SOFIA will normally take off and land at the same airport, so the aircraft will trivially be in range of the landing airport for at least half the flight.

We change the call to Feasible() as follows: first, we find the shortest dead leg that enables the observation for the desired duration. If the aircraft can return to the landing airport after completing *both* this dead-leg and the observation, then the observation is feasible, otherwise it is not feasible. We next postpone the check until *after* deciding to add an observation to the flight plan (after the step Extend F by e in Figure 2.) If the aircraft can't return to the landing airport after performing the chosen observation and its shortest enabling dead-leg, then it is discarded and another observation is chosen to extend the flight. This will reduce the expected number of checks significantly when most observations are feasible. Some observations previously considered feasible will not pass this check, and so the set of flight plans that can be produced by the algorithm is *restricted*. However, this restriction will reduce the value of the flight plans found only when *high priority* observations are excluded late in the flight. In practice we find comparable flight plans after making this modification with a modest increase in speed. In the interests of brevity, we do not report these results.

A Condition for the Shortest Dead-Leg

Even after reducing the number of checks to ensure that the aircraft can land, brute force search is still required to find the shortest dead-leg that enables an observation. However, we can take advantage of the new restricted feasibility condition by defining a function whose zeros correspond to the shortest dead-leg enabling the observation. This subproblem can be efficiently solved by using zero-finding algorithms such as Newton's Method. Because the resulting formulation allows us to search the full continuous space of dead-legs, we avoid discretizing the search space to enable brute-force search, and may also find *shorter* dead-legs.

Using the new conditions on object feasibility, the dead-leg construction phase of the feasibility check requires finding the heading and duration of the shortest dead-leg that enables the observation for a sufficient amount of time. A dead-leg may be necessary for one of two reasons: an observation is not visible at the current position and time, or an observation is not visible for long enough. We will treat these cases separately.

Consider the *feasible region* of an observation o at time θ. This region is the set of positions on the Earth from which the observation is visible, and is the annulus defined by two circles centered at the nadir position of o whose radii are the *coelevation* limits of the telescope (in SOFIA's case, the radii of these circles are 30° and 70°). Suppose the aircraft position $\mathbf{p}$ is outside the feasible region at time θ. We want the aircraft to be in the feasible region after completing the dead-leg, either by making the object rise or set by changing position. Now, the shortest leg would put the aircraft on the *boundary* of the feasible region, as opposed to anywhere strictly inside it. This means that the object elevation h will equal one of the two elevation limits after flying the dead-leg. If the aircraft begins inside the inner circle of the annulus, then we want the object to be precisely at the the upper telescope elevation limit of 60°, while if it is outside the outer circle, we want the object to be at the lower telescope elevation limit of 20°.

If the object was fixed relative to the ground, flying directly towards or away from the object would maximize the rate of change of the object elevation. However, the object

(and therefore the feasible region) appears to move across the Earth as the Earth rotates. A dead-leg that tracks the object as it moves would not minimize the flight distance. We use the following intuition: we search for a dead-leg that *ends* with the aircraft flying either directly towards or directly away from the object to be observed. Intuitively, this is the correct policy when the object is nearly in view, or near the end of longer dead-legs. Observatory policy will normally prevent dead-legs longer than a few tens of minutes, so this intuition will likely produce very short, if not "locally optimal" dead-legs.

Suppose flying a dead-leg with initial heading b for duration d results in aircraft heading vector $\vec{\mathbf{V_d}}$ at the aircraft's new position. The angle r between $\vec{\mathbf{V_d}}$ and the object azimuth at the new position $\vec{\mathbf{T_P}}$ is given by:

$$r = \cos^{-1}\left(\frac{\vec{\mathbf{V_d}}\vec{\mathbf{T_P}}}{||\vec{\mathbf{V_d}}||\,||\vec{\mathbf{T_P}}||}\right) \quad (4)$$

Thus, we have the following problem: find b, d such that $F_1(b,d) = \langle f_1(b,d), f_2(b,d)\rangle = \langle 0,0\rangle$ where $f_1(b,d) = r$ i.e. the difference between the object azimuth and the final heading of the aircraft after flying the dead-leg defined by b, d, and $f_2(b,d) = e - h$ is the difference between the final object elevation and the telescope elevation limit e closest to the initial object elevation.

Now consider the case where the object violates the elevation limits at some point during the observation, regardless of whether or not it is initially visible. We see that the flight track exits the annulus (and possible re-enters it later on). In this case, we can set up a function very similar to that we used when the observation was initially outside the feasible region. We now want to find b, d such that $F_2(b,d) = \langle f_1(b,d), f_3(b,d)\rangle = \langle 0,0\rangle$, where $f_3(b,d)$ is the difference between the *extreme* object elevation achieved during the flight-leg and the telescope elevation limit violated during the observation. The intuition behind this is that the dead-leg we wish to fly should just barely nudge the ground track of the flight-leg inside the feasible region. f_1 remains the same. Figure 5 shows a situation in which we would zero F_2 while searching for a dead-leg. Initially, the aircraft could not observe the object without the elevation exceeding the upper elevation limit, whose boundary is shown. However, it is possible to fly a 10 minute dead-leg to a new position, from which the maximum elevation achieved during the flight-leg does not exceed the elevation limits.

Unlike the previous case, where we only needed to compute the heading and object elevation at the end of the dead leg, we now must find either the minimum or maximum of the elevation over the course of the flight-leg. We perform binary search over the ground track to find the extreme of the object elevation.

In both cases, we have now reduced the problem of finding the shortest dead-leg to the problem of finding a zero of a function, which can be solved efficiently using a variety of methods as long as F satisfies some simple conditions.

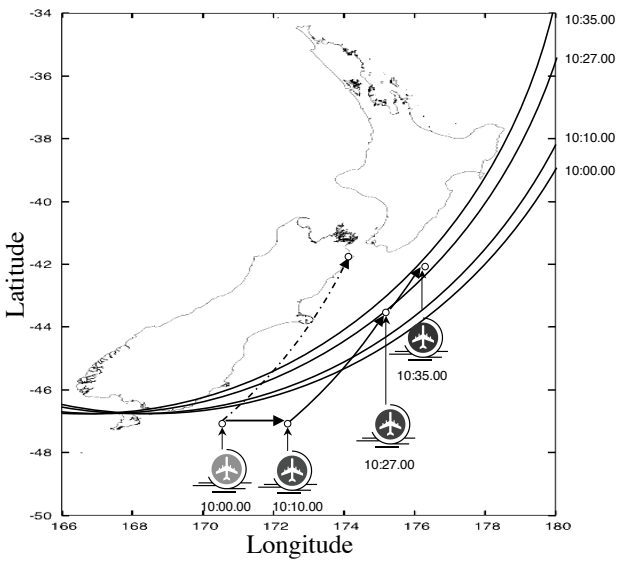

Figure 5: Flying a short dead-leg to enable an observation. The feasible region boundary shown is the upper elevation limit. In this case we would zero F_2 to search for the dead-leg enabling this observation. The aircraft's initial location is shown at 10:00:00. The dead-leg lasts 10 minutes, after which the flight-leg begins. At 10:27:00 the object elevation achieves a maximum; the figure also shows the feasible region at 10:27:00, and shows that the elevation limits are not violated by the flight-leg. The flight leg ends at 10:35:00.

Properties of the dead-legs

The behavior of zero-finding algorithms depends on how many zeros F has and how they are distributed. Also, the resulting dead-legs may not be feasible given other constraints on how the aircraft flies. We now analyze the zeros of the functions F_1 and F_2 and their corresponding dead-legs.

First of all, we observe that there are a countably infinite number of zeros of both F_1 and F_2. This does not pose a serious problem; these zeros are widely separated, and some require that the aircraft fly all the way around the world multiple times. The dead-leg duration restriction imposed by the ForwardPlanner algorithm will eliminate long dead-legs. However, Newton's Method might not find the shortest dead leg, and either incorrectly conclude that some observation is not feasible or return a suboptimal dead-leg.

Also, not all zeros correspond to valid dead-legs. For example, a dead leg whose duration is negative is impossible for the aircraft to fly. Also, short dead-legs may violate the minimum turn duration of the aircraft. A standard rate turn for a 747 is 180 degrees in 2 minutes. If the heading change and duration of the dead-leg violate this constraint, then the minimum dead-leg is impossible to achieve, but a longer dead leg might enable the leg. Under these circumstances, Newton's Method would incorrectly report that an observation is infeasible. Despite these potential drawbacks, this approach imposes no limitations on the heading or durations

of the dead-legs, so it might find dead-legs that could not be found by discretizing dead-leg durations and headings.

Finding dead-legs By Zeroing

In this section we will describe how to find dead-legs by zeroing F_1 and F_2.

Newton's Method and Cramer's Rule

Newton's Method is our choice for finding the zeros of F_1 and F_2. It is simple to implement and very fast (Gill, Murray, & Wright 1981). Newton's Method requires an initial guess for the zero; let this be denoted b_0, d_0 with future iterates denoted b_i, d_i . For functions F of 2 inputs and 2 outputs, the method proceeds as follows:

1. Compute $F(b_i, d_i) = \langle f_1(b_i, d_i), f_2(b_i, d_i)\rangle \equiv \langle f_1, f_2\rangle$
2. Compute the Jacobian (matrix of partial derivatives):

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial b}(b_i, d_i) & \frac{\partial f_1}{\partial d}(b_i, d_i) \\ \frac{\partial f_2}{\partial b}(b_i, d_i) & \frac{\partial f_2}{\partial d}(b_i, d_i) \end{pmatrix} \equiv \begin{pmatrix} p & q \\ r & s \end{pmatrix}$$

3. Compute the determinant of $J : |J| = ps - qr$. If this is smaller than error tolerance t then set $|J| = t$ (preserving the sign of $|J|$).
4. Compute the Cramer's Rule update: $db = \frac{f_2 q - f_1 s}{|J|}$ and $dd = \frac{f_1 p - f_2 r}{|J|}$
5. Set $b_{i+1} = b_i + db$ and $d_{i+1} = d_i + dd$
6. If $\langle b_{i+1}, d_{i+1}\rangle \approx \langle 0, 0\rangle$ or step limit reached, then halt, otherwise go to step 1.

Directly calculating the derivatives of the functions F_1 and F_2 is difficult because of the gridded wind model that influences the ground track, which in turn influences the elevation (remember, the elevation is a function of time and position). Consequently, we use finite differencing to compute all of our derivatives numerically (Gill, Murray, & Wright 1981). Zeroing F_1 only requires constructing the dead-leg preceding an observation, because evaluating F_1 only requires the heading and the object elevation at the end of the dead-leg. Zeroing F_2 requires constructing both the dead-leg and the flight-leg, because evaluating F_2 requires the extreme elevation of the object during the flight-leg. Of the available schemes, we chose forward differencing over centered differencing because of the smaller number of function evaluations required. Forward differencing requires evaluating F a total of 3 times to compute the partial derivatives. Thus, zeroing F_1 requires 3 dead-leg constructions per step, and zeroing F_2 requires 3 dead-leg constructions and 3 flight-leg constructions per step.

The Initial Guess

Algorithms like Newton's Method are highly sensitive to the closeness of the initial guess to the actual zero of the function. Newton's Method has *quadratic convergence* near a zero, which (roughly) means that the number of correct digits in the guesses doubles at each step. The brute-force dead-leg search does a blind search over possible headings and durations, so the number of correct digits in each guess improves by only a constant factor (at best) each step. Thus, using this methodology to find dead-legs should be an obvious performance win. However, we must make good initial guesses to benefit from rapid convergence.

Guessing the initial heading requires determining how an object's elevation is changing, and choosing the flight direction to make the elevation change correctly. Guessing the initial dead-leg duration requires estimating the change in elevation that the dead-leg must achieve, and then estimating the rate of change of the elevation during the dead-leg. Suppose that either the object is below the lower elevation limit and rising or above the elevation limits and setting. In this case we zero F_1. If the target is initially too high we want it to set faster. In this case we want to fly away from it, i.e. we guess $b_0 = A_f - 180°$. Similarly, if the object is initially too low, we want it to rise faster, so we want to fly towards it, i.e. we guess $b_0 = A_f$. Now suppose the object is moving out of the feasible region. In these cases we zero F_2. If the object initially is rising, either it will rise continuously or eventually set. We want to make it rise slower initially, set faster later, or both. In any case, we want to fly away from the object, so we guess $b_0 = A_f - 180°$. If the object is initially setting, either it continually sets or sets then rises; we either want it to set slower, rise faster, or both. In any case, we want to fly towards the object, so we guess $b_0 = A_f$. Only at high latitudes is it possible for an object to move into and then out of the feasible region; under these circumstances, we zero F_2. We might also find after successfully zeroing F_1 that we must zero F_2.

Guessing the duration is somewhat more complex. Calculating the maximum required change in elevation Δh at the current position and time is simple once we have calculated the test-leg. However, we have to account for the rate of change of the elevation both as a function of time, and the change in position as the aircraft flies. Define r_e as the equatorial radius of the Earth, ϕ_p as the latitude component of the aircraft's location $\mathbf{p}$, and v as the aircraft's estimated ground-speed. We compute the instantaneous vectors of the aircraft's ground speed and Earth's rotation, then use the law of cosines to determine the aggregate effect on the object elevation, resulting in the following guess: $v_{rot} = \frac{2.0\pi r_e}{24.0\phi_p}$ and $d_0 = \frac{\Delta h r_e}{\sqrt{v^2 + v_{rot}^2 + 2.0 v \sin(b_0)}}$.

Matters of Convergence

Newton's Method depends on the function being zeroed to obey some properties to guarantee convergence. Our functions do not obey these properties all of the time, and so Newton's Method occasionally fails to converge.

Newton's Method is "non-local", in the sense that it can generate any point in $\Re^2$ during any step. Thus, if F is not defined on every element of $\Re^2$, Newton's Method may fail to converge to a solution even if one exists. F_1 and F_2 are not well defined for sufficiently short or long dead-leg durations. The problem with long durations is due to the built-in nature of the fuel model. Essentially, if a Newton step requires the aircraft to fly long enough that it would run out of fuel, we can't evaluate the ground track of the flight-leg. The problem with short durations has been explained above. Thus, convergence of Newton's method may be interrupted if any intermediate step violates one of these conditions. This is

a problem because it is conceivable that the zero found by Newton's method can correspond to a legitimate supporting dead-leg even if an iteration of Newton's method corresponds to an impossible dead-leg. If the function or the derivatives can't be evaluated during Newton's Method, our only option is to truncate the feasibility check and report that the observation is not feasible. Additionally, we could find Newton's Method failing to converge after a large number of steps; we thus use a cutoff value to terminate search. In practice, we found few instances when Newton's Method could not establish the feasibility of an observation and using the brute-force approach could do so.

Empirical Results

In this section we describe experiments designed to test the value of using Newton's Method to speed up the feasibility check for ForwardPlanner.

Sample Problems

We used 24 of the problem instances described in (Frank & Kürklü 2003) to determine the utility of our new techniques. We restricted our attention to the "Single Day" instances; these contain between 6 and 11 observations. The priorities of all observations are identical, and all observations can be scheduled on a designated flight day. Thus, the principal goal is to find an efficient flight with all of the observations scheduled. The amount of lookahead influences the number of leg-construction steps required to compute the heuristics; for these experiments we set the lookahead depth to 4. The maximum dead-leg duration was set to 4 hours. For the brute-force search, we used a dead-leg duration increment of 1 minute and a heading increment of 7.5°. For Newton's Method we used a step cutoff of 150 and error tolerance $t = 10^{-6}$. The step parameters used in forward differencing were: $s_1 = 0.01^\circ$ and $s_2 = 60$ seconds. Experiments were run on a Sun Workstation with dual 600 MHz CPUs and 2048 Mb memory. The aircraft takeoff weight was fixed at $210,000$ pounds of fuel for all flights; this typically gives about 10 hour flights. The altitude was fixed at $35,000$ feet. We ran ForwardPlanner 20 times with each of the two feasibility methods. We compare the CPU times and the best flight efficiency results for the cases in which both approaches found a plan with all observations scheduled.

Figure 6 shows the average time to construct a single flight plan, contrasting the brute force and Newton's method driven feasibility checks. The speedups realized when using Newton's Method to check feasibility are dramatic for all but a small number of very short (4 observations) flight plans. In only 4 instances was either version of ForwardPlanner unable to schedule all observations to establish feasibility. Figure 7 shows the maximum efficiency of the 20 flight plans constructed for the cases where both versions of ForwardPlanner were able to schedule all observations. We see that the efficiencies are quite comparable; thus, eliminating solutions has not resulted in worse flight plans.

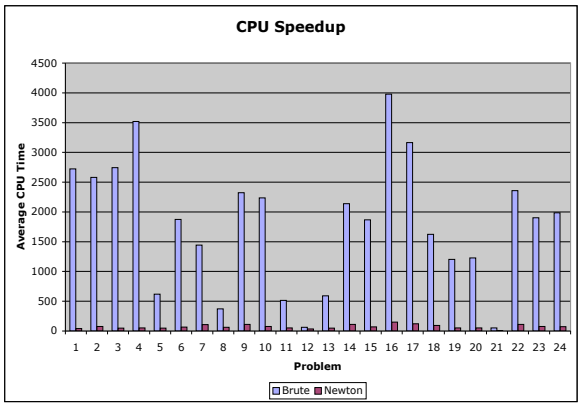

Figure 6: Performance improvements.

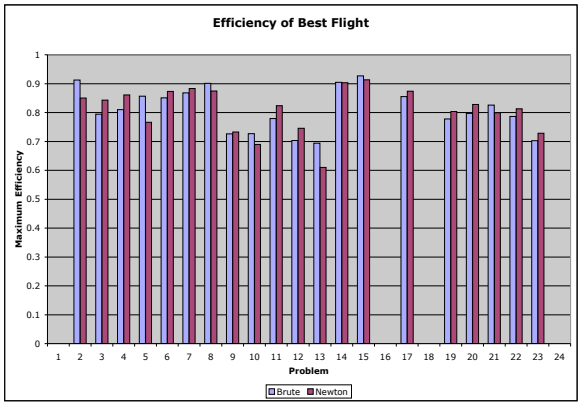

Figure 7: Efficiency of the best flights.

To see how the reduction in the number of legs corresponded to computational improvements, we analyzed a small number of flights from each of the takeoff airports. We compared the number of flight-legs and dead-legs constructed using the brute-force dead-leg search approach and Newton's Method for establishing feasibility over 20 runs. The results are shown in Figure 8. We see that the number of leg construction steps is dramatically reduced, leading consistently to increased speed. The speedup factor is typically smaller than the reduction in the number of legs constructed due to differences in the time required to construct flight-legs versus dead-legs, and added overhead resulting from the rest of Newton's Method such as the search for the minimum or maximum elevation required when zeroing F_2.

Problem	Flight-legs (B)	(N)	Dead-legs (B)	(N)	CPU Speedup
1	366,416	3,824.5	365,108	5,221.55	68x
2	326,840	5,854.4	324,890	5,577.75	40x
3	326,396	5,078.35	323,615	4,337.6	31x
5	66,391.1	2972.45	64,877.1	1737.6	13x
6	222,304	4,242.5	223,116	6,158.9	29x

Figure 8: Comparison of Newton's Method and Brute Force method of establishing observation feasibility on a small set of sample problems.

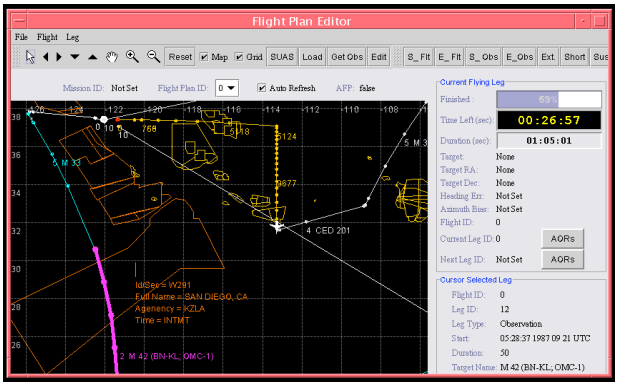

Figure 9: The SOFIA Flight Management System GUI.

SOFIA Flight Management

The SOFIA Flight Management System shown in Figure 9 is a software tool suite designed to facilitate the planning of flights. ForwardPlanner is fully integrated with all other software tools used to manage the flight plannning process for SOFIA. Users can construct inputs for SFPPs by assembling lists of observation requests and selecting flight days. Flight plans produced by ForwardPlanner can be displayed and superimposed on a variety of different maps, including Special Use Airspace (SUA) boundaries. The graphical display includes pop-ups with information about the observation and the aircraft performance during the indicated leg. The user can also display lists of scheduled and rejected observations in summary tables. Finally, users can simulate flight plans in a number of ways to test their robustness to uncertain conditions.

Conclusions and Future Work

We have described an application of AI techniques to the problem of scheduling astronomy observations on an airborne telescope. The resulting problem has complex interacting constraints, discrete and continuous decisions, as well as competing optimization criteria. We reduced the number of expensive observation feasibility checks by a novel combination of well-founded approximations and continuous optimization. Despite the fact that these assumptions eliminate some feasible flight plans, the new algorithms dramatically increase the speed of the algorithm at little cost in terms of the value of the flight plans produced. The resulting increase in speed improved the overall performance of the original heuristic-driven stochastic sampling approach. This approach can be employed for other planning and scheduling problems with mixtures of discrete and continuous variables. By analyzing the nature of the continuous decisions, it may be possible to reduce the set of options for these decisions to the solutions of an optimization problem that can be solved efficiently.

There are additional constraints that must be accounted for by ForwardPlanner prior to deployment. First, the aircraft must not cross any Special Use Airspace (SUA) zones. While these zones are relatively sparse, they may generally force longer dead-legs in flight plans. More importantly, the observation requests will have constraints on line-of-sight water vapor that must also be satisfied. The existing weather predictions will include estimated water vapor, which further constrains the aircraft's trajectory, and will also generally lead to longer dead-legs. Handling these constraints will impose further requirements on the feasibility check, potentially requiring new approaches to ensure that good flight plans can be produced quickly.

Acknowledgments

We would like to thank Karen Gundy-Burlet and the anonymous reviewers for their comments and insights. We would also like to thank Lan Lin for the Flight Management screen shot. This work was funded by the SOFIA Projects Office and by the NASA Intelligent Systems Program.

References

Bresina, J. 1996. Heuristic-biased stochastic sampling. In *Proceedings of the 13th National Conference on Artificial Intelligence*.

Ferziger, J. 1981. *Numerical Methods for Engineering Applications*. John Wiley and Sons.

Frank, J., and Kürklü, E. 2003. Sofia's choice: Scheduling observations for an airborne observatory. *Proceedings of the* 13^{th} *International Conference on Automated Planning and Scheduling*.

Gill, P.; Murray, W.; and Wright, M. 1981. *Practical Optimization*. Academic Press.

Johnston, M., and Miller, G. 1994. Spike: Intelligent scheduling of the hubble space telescope. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling*. Morgan Kaufmann Publishers.

Meeus, J. 1991. *Astronomical Algorithms*. Willmann-Bell, Inc.

Potter, W., and Gasch, J. 1998. A photo album of earth: Scheduling landsat 7 mission daily activities. In *Proceedings of the International Symposium Space Mission Operations and Ground Data Systems*.

Smith, D. 2004. Choosing objectives in over-subscription planning. *Proceedings of the* 14^{th} *International Conference on Automated Planning and Scheduling*.